Remarks:

Reconsideration of the application, as amended herein, is respectfully requested.

Claims 13 - 25 are presently pending in the application. Claims 13, 20 and 22 have been amended. Claims 1 - 12 were previously canceled.

In item 2 of the above-identified Office Action, claims 13 - 15, 18 - 22 and 25 were rejected under 35 U.S.C. § 103(a) as allegedly being obvious over U. S. Patent No. 6,434,524 to Weber ("**WEBER**") in view of U. S. Patent No. 5,678,039 to Hinks et al ("**HINKS**"). In item 3 of the Office Action, claims 16, 17, 23 and 24 were rejected under 35 U.S.C. § 103(a) as allegedly being obvious over **WEBER** in view of **HINKS**, and further in view of United States Patent No. 6,708,164 to Cseri et al ("**CSERI**").

Applicant respectfully traverses the above rejections, as applied to the amended claims.

More particularly, Applicant's claims 13 and 19 have been amended to recite, among other limitations:

> **checking each dialog element** of a user interface of the computer program to determine **whether a character string present in the respective dialog element includes a wildcard character string**; and

> **finding** in the text memory identification expressions associated with the wildcard character string contained in the computer program **and replacing the wildcard character string in the computer program with the associated message character strings in the text memory**, [emphasis added by Applicant]

Applicant's independent claim 20 has been amended to recite similar limitations, among others.

As such, Applicant's claims require, among other things, that the program checks whether a character string of a dialog element **of the user interface** includes a wildcard character string and, if so, finds message character string associated with the particular wildcard character string to replace the latter. The amendments to Applicant's independent claims are supported by the specification of the instant application, for example, page 8 of the instant application, line 31 - page 9, line 12, which state:

> **FIG. 1 shows a dialog box in a user interface of a computer program with a text field 1, a first button 2 and a second button 3.** This dialog box may have been created, by way of example, using ordinary program libraries for creating graphical user interfaces (GUI libraries), for example by programming or by using a development tool for a computer-aided development of user interfaces. The text usually contained in the popular graphical elements, in this figure the text which is contained in the text field 1, in the first button 2 and in the second button 3, has usually been attributed as a character string parameter (for example "string"), as in the aforementioned development methods for user interfaces.

> **In this example, these character strings are in the form of wildcard character strings** which start with a characteristic prefix, for example two successive paragraph characters. In this way, wildcard character strings can easily be distinguished from dialog texts, which are not wildcard character strings, in the subsequent method. [emphasis added by Applicant]

See also, for example, page 10 of the instant application, line 16 - page 11, line 2, which states:

> Having been initiated by this function call, **each dialog element 1, 2 and 3 in FIG. 1 is now successively visited during the further program/method execution, a check is performed to determine whether the character string which is present in the respective dialog element is a wildcard character string** by looking for the characteristic prefix ("§§"), and then the characteristic prefix is removed from the respective wildcard character string and the remaining XML path is used to read the entry addressed by this path in the XML file, which has already been opened during production of the language handler object. After that, the value associated with the entry, namely the message character string, is substituted for the character string originally contained in the respective dialog element, i.e. the wildcard character string stored in the respective dialog element is replaced with the relevant message character string which has been ascertained. Within this context, it is possible for a single path stored as a wildcard character string in the respective dialog element to replace a plurality of associated character string values, too, for example ToolTip texts (explanatory texts which pop up on the basis of the position of a mouse pointer on the display panel) associated with the dialog elements 1, 2 and 3, or status bar texts. [emphasis added by Applicant]

Applicant's invention is useful in customizing the user interface to a particular person or location (i.e., "localization"). See, for example, page 14 of the instant application, lines 11 - 26, which state:

> In summary, the computer system 11 shown in FIG. 5 is thus, in the general sense, an electrical appliance with at least one microprocessor 12 and a display apparatus 13 on which at least one display object 14a, 14b is shown in a starting language. A selectable text memory chip 15 is provided which contains alphanumeric message character strings in the selection language which are associated with alphanumeric identification expressions. **To change from the starting language to the selection language which is to be displayed from now on, this chip outputs message character strings in the selection language which are associated with selected identification expressions, upon request by the microprocessor 12, when an identification expression is applied to it which corresponds to a wildcard character string associated with the at least one display object 14a, 14b**. [emphasis added by Applicant]

However, the prior art cited in the Office Action fails to teach or suggest, among other limitations of Applicant's claims, **checking whether a character string of the user interface is a wildcard character string and, if so, finding an associated message character string, stored in memory, to replace the wildcard character string with the associated message character string.**

More particularly, page 2 of the Office Action alleged that col. 9 of **WEBER**, lines 42 - 63, and Fig. 2 of **WEBER** disclose finding text memory identification expressions associated with wildcard character strings and replacing the wildcard character strings with the text memory identification expressions. Applicant respectfully disagrees that **WEBER** discloses Applicant's presently claimed checking, finding and

replacing steps (claims 13 and 19) or means for checking, for finding and for replacing (claim 20). Rather, **WEBER** merely discloses an object interactive user interface using speech recognition and natural language processing. Col. 9 of **WEBER**, lines 42 - 63, state:

> The flow continues to block 330 **where certain "word-variables" are replaced with an associated wildcard function by variable replacer** 204 in preparation for accessing the NLP database 218. As used herein, the term "word-variables" refers to words or phrases that represent amounts, dates, times, currencies, and the like. For example, in one embodiment the phrase "what movies are playing at 8 o'clock" would be transformed at block 330 to "what movies are playing at $time" where "$time" is a wildcard function used to represent any time value. As another example, in one embodiment the phrase "sell IBM stock at 100 dollars" would be transformed at block 330 to "sell IBM stock at $dollars" where "$dollars" is a wildcard function used to represent any dollar value. This act may be accomplished by a simple loop that searches the phrase for key tokens such as the words "dollar" or "o'clock" **and replaces the word-variables with a specified wildcard function**. In order to keep track of the location in the phrase where the substitution was made, an array memory be used. This allows re-substitution of the original word-variable back ubti [sic] the phrase at the some position after the NLP database 218 has been searched. [emphasis added by Applicant]

As such, the text of **WEBER** cited in the Office Action merely describes **how specific word variables input by the user are replaced with a corresponding wild card.** Thus, certain numeric details, (i.e., "**8** o'clock", "**100** dollars") found in a sentence to be analyzed are replaced with a general variable

(i.e., "$time", "$dollars"). This, however, neither teaches, nor suggests, the process claimed by Applicant.

In particular, Applicant's claims require, among other things, that the **character string of dialog expressions of the user interface be checked for the presence of wild card character strings, which are then replaced with associated message character strings**, whereas, in accordance with **WEBER, concrete character strings entered by the user are replaced with wild card character strings**. In other words, **WEBER** specifically teaches **exchanging a concrete message string with a wild card character string**, which entirely contradicts, and specifically teaches away from, Applicant's claimed invention of replacing a wild card character string with an associated message string.

Later, in **WEBER**, the wildcard **inserted by the variable replacer 204** of **WEBER** can be changed back to the original word variable after the NLP database of **WEBER** has been searched. However, this wildcard of **WEBER** is **not part of a dialog element of a user interface** of the computer program, as required by Applicant's claims. Rather, this wild card was inserted into an input character string by the variable replacer 204 of **WEBER**. As such, **WEBER** specifically teaches

away from Applicant's claimed invention of checking a dialog expression **of the user interface** for a wildcard string.

Further, **WEBER fails** to teach or suggest, among other limitations of Applicant's claims, **a text memory containing expressions associated with wildcard character strings that can be present in the dialog expressions of the user interface**. In contrast, **WEBER** contains an array in which **an expression, originally entered by the user**, is stored, after being replaced with a wildcard by the variable replacer of **WEBER**. As such, the memory array of **WEBER** cannot be analogized to Applicant's particularly claimed **text memory**.

The **HINKS** reference, cited in the Office Action in combination with **WEBER**, additionally fails to teach or suggest the above-discussed limitations of Applicant's claims, among others.

More particularly, **HINKS** discloses a system and method for translating software into localized versions. In contrast to Applicant's claimed invention, **HINKS** also fails to teach or suggest, among other limitations, checking each **dialog element of a user interface** of the computer program to determine whether a character string present in the respective dialog element **includes a wildcard character string**. Rather, **HINKS** discloses using a "software translation kit" (STK) of **HINKS**

that analyzes the software program to be translated and, then, **provides human translators with the respective text passages to be translated.** See, for example, col. 8 of **HINKS**, lines 6 - 13. The human translators translate the questionable text passages into another language. Afterwards, the translated texts are again bound into the localized software. See, for example, col. 8 of **HINKS**, lines 14 - 16. As such, in **HINKS**, each dialog box is **not checked for a wildcard character string**, as required by Applicant's claims.

Further, Applicant's claim 13 requires, among other limitations:

> carrying out the finding and replacing steps **during a runtime** of an executable binary computer program; [emphasis added by Applicant]

Similarly, Applicant's independent claim 19 recites, among other limitations:

> running the computer program by runtime execution of an executable binary program file and, **during the execution**:
>
> **finding** in the text memory identification expressions associated with the wildcard character string contained in the computer program **and replacing** the wildcard character string in the computer program with the associated message character strings in the text memory, by assigning the message character strings to memory variables in the running computer programs. [emphasis added by Applicant]

Additionally, Applicant's claim 20 recites, among other limitations:

> the computer program being in executable binary code and **said means for finding and replacing being contained in the computer program.** [emphasis added by Applicant]

As such, all of Applicant's claims require that, among other things, an identification expressions associated with the wildcard character string be found in a text memory **and** the wildcard character string be replaced with the found associated message character strings **during runtime execution of the computer program in which localization is occurring**. This is clearly not the case in **HINK**. Rather, replacement of the phrases to be translated in **HINK** is not performed during the run time of the computer program requiring localization. This can be seen, for example, from col. 3 of **HINK**, lines 35 - 43, which state:

> **Once the translated resource file has been generated, the target product is rebuilt with the new sources.** The file may be simply stored back with the source files, as a translated resource file(s); or, the translated resource file may be compiled and bound back into the target program directly. In either instance, the underlying program code (i.e., the code which the programmer has written to carry out the functionality of the program) has remained untouched by the process. [emphasis added by Applicant]

Additionally, among other limitations of Applicant's claims, **HINKS** fails to teach or suggest, a text memory containing

message character strings assigned to corresponding identification expressions. Further, because **HINKS** does not disclose checking dialog elements wildcard character strings, there cannot be any **runtime** replacement of wild card character strings with associated message character strings from a text memory, as required by Applicant's claims.

For the foregoing reasons, among others, Applicant's claims are believed to be patentable over **WEBER** even in view of the **HINKS** reference.

The **CSERI** reference, cited in item 3 of the Office Action, in combination with **WEBER** and **HINKS** against certain of Applicant's dependent claims, does not cure the above-discussed deficiencies of the **WEBER** and **HINKS** references. As such, for the foregoing reasons, among others, Applicant's claims are believed to be patentable over **WEBER**, **HINKS** and **CSERI**, whether taken alone, or in combination.

It is accordingly believed that none of the references, whether taken alone or in any combination, teach or suggest the features of claims 13, 19 and 20. Claims 13, 19 and 20 are, therefore, believed to be patentable over the art. The dependent claims are believed to be patentable as well because they all are ultimately dependent on claims 13 or 20.

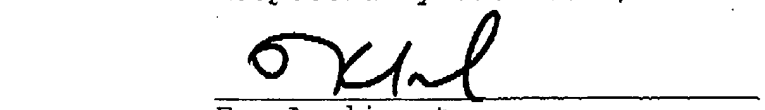
In view of the foregoing, reconsideration and allowance of claims 13 - 25 are solicited.

In the event the Examiner should still find any of the claims to be unpatentable, counsel would appreciate receiving a telephone call so that, if possible, patentable language can be worked out.

If an extension of time for this paper is required, petition for extension is herewith made.

Please charge any fees that might be due with respect to Sections 1.16 and 1.17 to the Deposit Account of Lerner Greenberg Stemer LLP, No. 12-1099.

Respectfully submitted,



For Applicant

Kerry P. Sisselman
Reg. No. 37,237

June 17, 2008

Lerner Greenberg Stemer LLP
Post Office Box 2480
Hollywood, FL 33022-2480
Tel: (954) 925-1100
Fax: (954) 925-1101